Applied Mathematical Modelling 38 (2014) 3975–3986

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Minimizing the total weighted late work in scheduling of identical parallel processors with communication delays

Foroogh Abasian [a], Mohammad Ranjbar [a,*], Majid Salari [a], Morteza Davari [b], Seyed Morteza Khatami [a]

[a] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Research Group for Operations Management, Faculty of Business and Economics, KU Leuven, Leuven, Belgium

ARTICLE INFO

Article history:
Received 26 August 2013
Accepted 23 January 2014
Available online 4 February 2014

Keywords:
Scheduling
Parallel processors
Communication delay
Branch-and-bound algorithm

ABSTRACT

This paper addresses a certain type of scheduling problem that arises when a parallel computation is to be executed on a set of identical parallel processors. It is assumed that if two precedence-related tasks are processed on two different processors, due to the information transferring, there will be a task-dependent communication delay between them. For each task, a processing time, a due date and a weight is given while the goal is to minimize the total weighted late work. An integer linear mathematical programming model and a branch-and-bound algorithm have been developed for the proposed problem. Comparing the results obtained by the proposed branch-and-bound algorithm with those obtained by CPLEX, indicates the effectiveness of the method.

1. Introduction

During the last two decades, the parallel processing has improved the performance of computing in many systems like real-time signal processing (Tokhi and Hossain [1]), image processing (Prajapati and Vij [2]) and robotic control (Jadud et al. [3]). Nevertheless, Ariosy et al. [4] show that the time restriction is usually an important factor in robotic control systems. In such systems the data, including a set of computational tasks, are collected using sensing devices and are processed in predefined time windows. Some tasks are precedence related using communication messages where each message carries certain amount of information. Usually, robots must react to particular programs on given due dates where each due date corresponds to a task. If the required information for a suitable reaction is not processed completely before or at a given time moment, the robot must react based on incomplete information. Obviously, the amount of gathered and processed data affects the accuracy of the control process, and all the information exposed after the given due date (called the information loss) is useless. The information lost is modeled as *late work* and should be minimized to increase the accuracy of the control systems.

In this paper, we consider the problem of scheduling a set of precedence related tasks on a target parallel system (Sinnen [5]), consisting of a set of identical processors connected by a communication network. In this system, each processor can execute only one task at a time and the execution is not preemptive. Also, the cost of communication between tasks executed

* Corresponding author. Address: Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, P.O. Box: 91775-1111, Mashhad, Iran. Tel.: +98 511 8805092.
E-mail addresses: foroogh.abasian@stu-um.ac.ir (F. Abasian), m_ranjbar@um.ac.ir (M. Ranjbar), msalari@um.ac.ir (M. Salari), morteza.davari@econ.kuleuven.be (M. Davari), sm_khatami@stu.um.ac.ir (S.M. Khatami).

http://dx.doi.org/10.1016/j.apm.2014.01.006
0307-904X/

on the same processor, local communication, is negligible and therefore considered zero. This assumption is based on the observation that for many parallel systems remote communication (i.e. interprocessor communication) is one or more orders of magnitude more expensive than local communication (i.e. intraprocessor communication). In addition, intraprocessor communication is performed by a dedicated communication subsystem. Interprocessor communication, referred to as *communication* hereafter, in the system is performed concurrently; there is no contention for communication resources. As an example the Uniform Memory Access (UMA), a shared memory architecture used in parallel computers, represents this model. For scheduling, suppose we are given a directed acyclic graph which represents the precedence relations among the tasks. The assumption is that for each task, a processing time, a due date and a weight corresponding to its relative importance, are given. The goal is to determine the best schedule, the processor and the start time corresponding to each task, in which the total weighted late work is minimized.

Using standard notification, which was originally presented by Graham et al. [6] and later extended by Veltman et al. [7], this problem can be shown as $P_m|prec,\ comu|Y_w$.

There are many papers related to the scheduling problems of identical processors in the literature such as Gendreau et al. [8] and Blazewicz et al. [9]. Verrite [10] considered the problem $P_m|prec,\ comu|T_{max}$ and $P_m|prec,\ comu|L_{max}$ where T_{max} and L_{max} indicate the maximum *tardiness* and the maximum *lateness*, respectively. Sinnen [5] reviewed different situations of task scheduling for parallel systems. Similarly, Drozdowski [11] considered scheduling problems for parallel processing.

Many research papers related to the minimization of *late work* are proposed in the literature. In particular, these papers can be classified into the five following categories: *single machine* (Potts and Van Wassenhove [12], Kovalyov et al. [13]), *parallel machine* (Blazewicz [14]), *flow shop* (Blazewicz et al. [15], Pesch and Sterna [16]), *job shop* (Blazewicz et al. [17]) and *open shop* (Blazewicz et al. [18]). For a comprehensive survey, interested readers are referred to (Sterna [19]). However, to the best of our knowledge, the problem $P_m|prec,\ comu|Y_w$ has not been taken into account in the literature.

As shown by Sterna [20], there is a polynomial time algorithm for $P_m|r_i,\ p_i = 1|Y_w$ while $P_2|p_j = 1,\ cains|Y$ is *NP*-hard. Since $P_2|prec,\ comu|Y_w$ is a generalization of $P_2|p_j = 1,\ cains|Y$, it is *NP*-hard as well.

The main contributions of this article are twofold: (1) we provide the first description of $P_m|prec,\ comu|Y_w$ and developed an integer linear formulation for it; (2) we develop a branch-and-bound (B&B) algorithm for the problem and derive upper and lower bounds as well as dominance rules to improve the performance of the B&B algorithm.

The remainder of this paper is organized as follows. Theory of the work including the mathematical statement of the problem and the proposed B&B algorithm are presented in Sections 2 and 3, respectively. Results and discussions are reported in Section 4. Finally, concluding remarks are presented in Section 5.

2. Problem statement

In $P_m|prec,\ comu|Y_w$, a set of tasks $N = \{1,\ldots,n\}$ must be processed on a set of identical parallel processors $M = \{M_1,\ldots,M_m\}$. For each task $i \in N$, we are given a processing time p_i, a due date d_i and a weight w_i where all parameters are supposed to be deterministic and non-negative integer values. Each task $i \in N$ must be processed without preemption on a processor. The output of some tasks constitutes the input of some others; thus, there is finish-to-start precedence relation between some pairs of tasks, represented by set A, i.e. a (strict) partial order on N. If s_i indicates the start time of task i, set A is defined as an irreflexive and transitive relation imposing the constraints $s_i + p_i + \Delta_{ik} \leqslant s_k$ for all $(i,k) \in A$ in which Δ_{ik} shows the communication delay between tasks i and k and is zero if both are processed on the same processor. We establish the directed acyclic graph $G(N,A)$ in which sets N and A correspond to the set of nodes and arcs, respectively. We aim to find a schedule that minimizes the total *weighted late work* where such a schedule can be obtained by employing efficient task partitioning and scheduling strategies. The late work of task i is mathematically defined as $LW_i = \min\{Tr_i,\ p_i\}$ where $Tr_i = \max\{f_i - d_i,\ 0\}$ indicates the tardiness of task i in which f_i shows the finish time of task i.

In order to formulate the problem, in the following we introduce some variables.

$$X_{ijt} = \begin{cases} 1; & \text{if task } i \text{ is completed on processor } M_j \text{ at time instant } t, \\ 0; & \text{Otherwise.} \end{cases}$$

$$Z_i = \begin{cases} 1; & \text{if} Tr_i \leqslant p_i, \\ 0; & \text{if} Tr_i > p_i. \end{cases}$$

The model reads as follows:

$$Min \sum_{i=1}^{n} w_i LW_i. \tag{1}$$

Subject to

$$Tr_i \geqslant tX_{ijt} - d_i; \quad \forall i \in N, \quad \forall j \in M \text{ and } t = 1,\ldots,T, \tag{2}$$

$$LW_i \leqslant Tr_i; \quad \forall i \in N, \quad \forall j \in M \text{ and } t = 1,\ldots,T, \tag{3}$$

$$LW_i \leqslant p_i; \quad \forall i \in N, \tag{4}$$

$$Z_i \geqslant \frac{p_i - Tr_i}{H}; \quad \forall i \in N, \tag{5}$$

$$Z_i \leqslant 1 - \frac{Tr_i - p_i}{H}; \quad \forall i \in N, \tag{6}$$

$$LW_i \geqslant Tr_i - H(1 - Z_i); \quad \forall i \in N, \tag{7}$$

$$LW_i \geqslant p_i - HZ_i; \quad \forall i \in N, \tag{8}$$

$$\sum_{j=1}^{m}\sum_{t=1}^{T} X_{ijt} = 1; \quad \forall i \in N, \tag{9}$$

$$\sum_{i=1}^{n} X_{ijt} + \sum_{\tau=1}^{T-t}\sum_{i|p_i>\tau} X_{ij(t+\tau)} \leqslant 1; \quad \forall j \in M \text{ and } \forall t = 1, \ldots, T, \tag{10}$$

$$\sum_{t=1}^{T} tX_{ijt} \leqslant H\left(1 - \sum_{t=1}^{T} X_{kjt}\right) + \sum_{t=1}^{T} tX_{kjt} - p_k; \quad \forall (i,k) \in A \text{ and } \forall j \in M, \tag{11}$$

$$\sum_{t=1}^{T} tX_{ijt} + \Delta_{ik}\sum_{t=1}^{T} X_{kj't} \leqslant \Delta_{ik}\sum_{t=1}^{T} tX_{kj't} - p_k + H\left(1 - \sum_{t=1}^{T} X_{kj't}\right) + H\left(1 - \sum_{t=1}^{T} X_{ijt}\right); \quad \forall (i,k) \in A \text{ and } \forall j,\ j'(j \neq j') \in M, \tag{12}$$

$$X_{ijt} \in \{0,1\}; \quad \forall i \in N, \quad \forall j \in M \text{ and } t = 1, \ldots, T, \tag{13}$$

$$Z_i \in \{0,1\}; \quad \forall i \in N, \tag{14}$$

$$Tr_i \in \mathbb{Z}^+; \quad \forall i \in N. \tag{15}$$

The objective function (1) is to minimize the total weighted late work. The constraint set (2) shows the tardiness of task i where tX_{ijt} indicates the finish time of this task and t is the time index. Also, T indicates an upper bound on the completion time of the project. If we assume all tasks are processed on a single processor, no communication delay will be applied and consequently we have $T = \sum_{i=1}^{n} p_i$. In the Appendix A, we prove that $T = \sum_{i=1}^{n} p_i$ does not eliminate the optimal solution. The relation $LW_i = \min\{Tr_i, i\}$ is formulated using constraints (3)-(8) in which H indicates a very big number. In fact, if $Tr_i \leqslant p_i$ then we must have $Z_i = 1$ to activate the constraint (8); hence, the constraint (7) will be redundant and we have $LW_i = Tr_i$. So, the constraints (4) and (5), by considering the values of Tr_i and p_i, determine the value of Z_i appropriately. The constraint (9) indicates that each task is completed only at a unique time and on a specified processor. The constraint set (10) is a set of forcing constraint and imposes that on each processor at most one task to be processed in a given time instant. Finish-to-start type precedence relations and communication delays among directly dependent tasks are formulated using the constraint (11) and (12). The rest of constraints define the domains of variables where $\mathbb{Z}^+$ indicates the set of non-negative integers.

3. A branch-and-bound algorithm

In this section, we develop a B&B algorithm to the resolution of the proposed problem. Particularly, the developed method is based on the depth-first strategy which works based upon assigning each task to different processors. Usually, each B&B algorithm includes two main schemes, i.e. *branching* and *bounding*. In the *branching* scheme, the B&B tree is constructed and different feasible schedules are investigated while in the *bounding* scheme, the goal is to fathom the nodes using suitable and efficient dominance rules.

3.1. The branching scheme

In our B&B, each schedule (solution) $\boldsymbol{s}$ is represented as $\boldsymbol{s} = (Pr(1), \ldots, Pr(n), s_1, \ldots, s_n)$ where $Pr(i)$ and s_i indicate the processor and start time of task $i \in N$, respectively. For each schedule in each level of the B&B tree, $Pr(i)$ and s_i are zero for unscheduled tasks. Moreover, let v_u^l be the u^{th} node in level l of the B&B tree in which an eligible task $i \in N$ i.e. a task that all of its predecessors are scheduled already, is scheduled on a processor $Pr(i) \in M$ in the earliest possible start time. This node corresponds to a partial schedule $\boldsymbol{s}$ in which values of some s_i and $Pr(i)$ for scheduled tasks have been determined. The B&B tree includes $n + 2$ levels where root node is placed in the highest level $(l = 0)$ and all complete schedules are

obtained in the lowest level ($l = n + 1$). Also, the number of offspring corresponding to node v_u^l equals to $|E(v_u^l)| \times m$ where $|E(v_u^l)|$ specifies the number of eligible tasks in node v_u^l, gathered in set $E(v_u^l)$. In the worst case in which $A = \varnothing$, we have $|E(v_u^l)| = n - l$ and consequently, at most $n!m^n$ schedules have to be investigated in total. Since the problem $P_m|prec, comu|Y_w$ is *NP*-hard, an exact algorithm with better than exponential time complexity is unlikely to exist. As a result, our developed B&B algorithm which implicitly enumerates the solution space has complexity of $O(n!m^n)$.

The branching scheme is illustrated on an example problem, shown in Fig. 1. This example includes $n = 10$ tasks and $m = 3$ identical processors in which the number inside each node gives the corresponding task number while the number along each arc displays the amount of communication delay. Finally, processing time, due date and weight corresponding to each task are listed in Table 1.

Fig. 2 demonstrates some different branching choices at levels $l = 1, 2$ and 3 of the B&B tree, in which each ordered pair $i - j$ represents the assignment of task i to the processor M_j. In this figure, all possible nodes of level $l = 1$ are depicted. Since there are numerous possible nodes at levels $l = 2$ and 3, only offspring of nodes 3–2 and 1–3 are shown in the second and third levels, respectively.

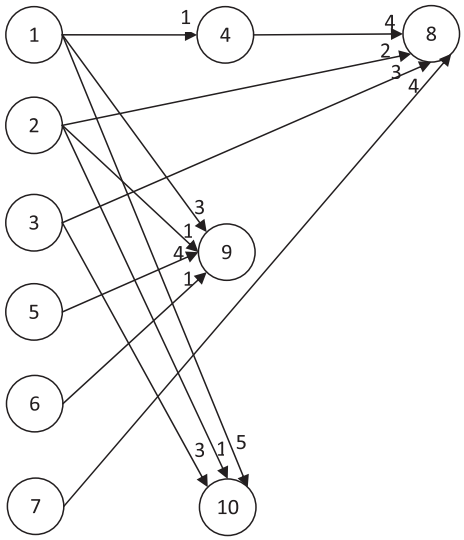

Fig. 1. The example problem.

Table 1
Data corresponding to the example problem.

Task	1	2	3	4	5	6	7	8	9	10
Duration	3	1	8	1	4	5	4	5	5	9
Weight	1	6	2	9	6	5	4	9	9	8
Due date	1	2	3	3	3	6	7	9	13	15

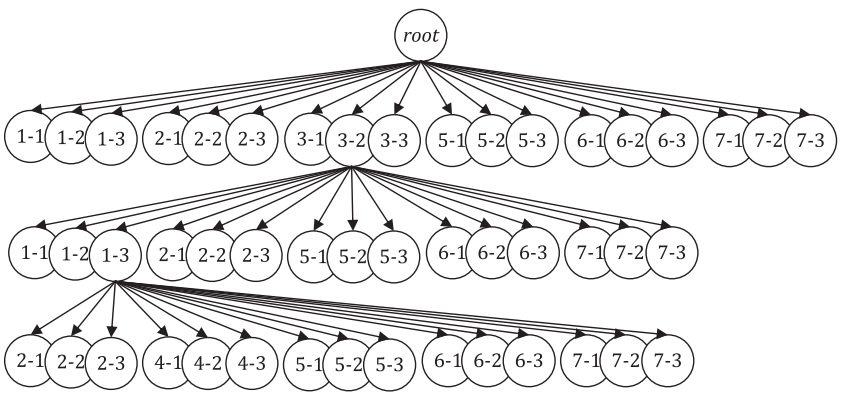

Fig. 2. Illustration of the branching scheme.

3.2. The bounding scheme

In this section, we propose some rules to obtain upper and lower bounds for the optimal objective value of $P_m|prec, comu|Y_w$. Moreover, some efficient dominance rules are developed to enhance the performance of the B&B algorithm.

3.2.1. Upper bound

In order to reach a fast algorithm to construct an initial feasible solution, to be considered as an upper bound (UB) for $P_m|prec,\ comu|Y_w$, we first construct a priority list of tasks, sorted based on the non-decreasing order of θ-values as follows: for each task $i \in N$, we define $Suc^+(i)$ as the set including task i and all (direct and indirect) of its successors and $\theta_i = \sum_{k \in Suc^+(i)} \frac{MWLW_k}{\delta_k + 1}$ in which $MWLW_k$ and δ_k represent the minimum weighted late work and the slack corresponding to task k, respectively. For each task k, we have $\delta_k = ls_k - es_k$ where ls_k and es_k indicate the latest and earliest start time of task k, respectively, and are calculated using the well-known critical path method (CPM) developed by Kelley and Walker [21].

It should be noticed that for each task k, $MWLW_k$ is calculated based on the earliest finish time of task k where the earliest finish times are calculated based upon the CPM without taking into account the communication delays.

According to the priority list, tasks are sequentially scheduled on the earliest possible start time until a feasible complete solution is obtained. In each iteration, the next task from the priority list is chosen and for that the earliest possible start time, in which communication delays are considered, is assigned such that no precedence constraint is violated.

For instance, consider the example problem for which the values of θ_i are reported in the second row of Table 2. Consequently, the priority list is constructed as $(3,1,2,10,4,7,8,5,6,9)$. Applying the described procedure to this example problem, we obtain finish times reported in the last row of Table 2. As a result, the weighted late work of the obtained schedule, depicted in Fig. 3, is 119.

3.2.2. Lower bound

Having a partial schedule $\boldsymbol{s}$ corresponding to $P_m|prec,\ comu|Y_w$, let ST be the set of scheduled tasks. A very simple lower bound (LB^1) for the partial schedule $\boldsymbol{s}$ is calculated by taking into account the weighted late work of scheduled tasks as follows:

$$LB^1 = \sum_{i \in ST} w_i \min(p_i,\ \max(0, f_i - d_i)). \tag{16}$$

On the other hand, corresponding to each partial schedule $\boldsymbol{s}$, it is possible to find a second lower bound, LB^2, for the weighted late work of unscheduled tasks (UT) which is given in (17). In particular, for each task $i \in UT$, maximum value of the earliest finish time (ef_i) must be calculated. Finally, combining LB^1 and LB^2 leads us to $LB = LB^1 + LB^2$.

$$LB^2 = \sum_{i \in UT} w_i \min(p_i, \max(0, ef_i - d_i)). \tag{17}$$

Table 2
Illustration of the initial solution procedure.

Task	1	2	3	4	5	6	7	8	9	10
θ_i	24.2	23.2	33.2	8.2	0.67	0	7.2	7.2	0	16
f_i	3	1	8	4	8	10	5	16	16	17

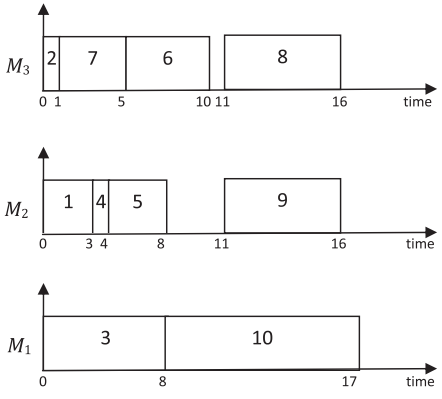

Fig. 3. Initial solution of the example problem.

Letting $ef_i = es_i + p_i$, in what follows, three rules are developed to calculate es_i. It should be noticed that in calculation of LB^2, for each $i \in ST$, we assume that $ef_i = f_i$. For each unscheduled task $i \in N$, let $pred^i_{v^l_u}$ represent the set of direct predecessors of task i at node v^l_u of the B&B tree, partitioned into two subsets scheduled tasks ($spred^i_{v^l_u}$) and unscheduled tasks ($upred^i_{v^l_u}$). In the first case, it is assumed that $|pred^i_{v^l_u}| = 1$, in the second case, the assumption is that $1 < |pred^i_{v^l_u}| \leqslant m$ while in the third case, it is presumed that $|pred^i_{v^l_u}| > m$.

- *Case* 1 ($|pred^i_{v^l_u}| = 1$): If $k \in spred^i_{v^l_u}$ and $Pr(i) = Pr(k)$, then $es_i = f_k$. Also, if $k \in spred^i_{v^l_u}$ and $Pr(i) \neq Pr(k)$, then es_i is calculated as $es_i = f_k + \Delta_{ki}$. Moreover, if $k \in upred^i_{v^l_u}$, then it is possible that in the optimal solution either $Pr(i) = Pr(k)$ or $Pr(i) \neq Pr(k)$. However, as a lower bound, we assume $Pr(i) = Pr(k)$ to ignore communication delay Δ_{ki}. Consequently, es_i is calculated as $es_i = ef_k$.
- *Case* 2 ($1 < |pred^i_{v^l_u}| \leqslant m$): In this case, es_i is calculated as $es_i = \max\{es^1_i,\ es^2_i,\ es^3_i\}$ in which es^1_i, es^2_i and es^3_i are formally described as follows:

$$es^1_i = \min_{k \in spred^i_{v^l_u}} \left(\max_{a \in spred^i_{v^l_u}, Pr(k) \neq Pr(a)} (f_k, f_a + \Delta_{ai}), \max_{b \in spred^i_{v^l_u}, Pr(k) = Pr(b)} (f_k, f_b) \right), \tag{18}$$

$$es^2_i = \min_{k,a \in upred^i_{v^l_u}} (\max(ef_k + \Delta_{ki}, ef_a), \max(ef_a + \Delta_{ai}, ef_k), ef_k + p_a, ef_a + p_k), \tag{19}$$

$$es^3_i = \min_{k \in spred^i_{v^l_u}, a \in upred^i_{v^l_u}} (f_k + p_a, \max(ef_a + \Delta_{ai}, f_k), \max(f_k + \Delta_{ki}, ef_a)). \tag{20}$$

A simple lower bound for tasks of subset $spred^i_{v^l_u}$ is $\max_{k \in spred^i_{v^l_u}} (f_k)$, while this bound could be significantly improved by taking into account the communication delays. To this aim, we assume the case in which $Pr(i) = Pr(k) \neq Pr(a)$ where $k, a \in spred^i_{v^l_u}$, then $es_i \geqslant \max_{a \in spred^i_{v^l_u},\ Pr(k) \neq Pr(a)} (f_k, f_a + \Delta_{ai})$. Also, for the case in which $Pr(i) = Pr(k) = Pr(b)$ and $b \in spred^i_{v^l_u}$, leads to $es_i \geqslant \max_{b \in spred^i_{v^l_u},\ Pr(k) = Pr(b)} (f_k, f_b)$. As a result, es^1_i is constructed as shown already in (18).Similarly, instead of the simple lower bound $\max_{k \in upred^i_{v^l_u}} (ef_k)$, we can construct es^2_i based on every two tasks $k, a \in upred^i_{v^l_u}$. Since processors of tasks $k, a \in upred^i_{v^l_u}$ are not determined yet, we consider four scenarios to construct es^2_i. In the first two scenarios, it is assumed that tasks k and a are processed on different processors while in the two other scenarios it is supposed that they are processed on the same processor. For instance, if $Pr(i) = Pr(a) \neq Pr(k)$, then $es_i \geqslant \max(ef_k + \Delta_{ki}, ef_a)$ and $es_i \geqslant \max(ef_a + \Delta_{ai}, ef_k)$ while if $Pr(i) = Pr(a) = Pr(k)$ then $es_i \geqslant \min(ef_k + p_a, ef_a + p_k)$. Thus, es^2_i is obtained by following what is given in (19). It is worth nothing that es^2_i could be improved by considering more than two tasks but it would be very time consuming.Finally, we develop es^3_i for tasks k and a where $k \in spred^i_{v^l_u}$ and $a \in upred^i_{v^l_u}$. Assuming $Pr(i) = Pr(k) = Pr(a)$ then $es_i \geqslant f_k + p_a$, while, if $Pr(i) = Pr(k) \neq Pr(a)$, then $es_i \geqslant \max(ef_a + \Delta_{ai}, f_k)$ and for the case in which $Pr(i) = Pr(a) \neq Pr(k)$, we have $es_i \geqslant \max(f_k + \Delta_{ki}, ef_a)$. Thus, es^3_i is constructed as shown in (20).Now, consider the partial schedule depicted in Fig. 4 corresponding to the example problem in which tasks 1 and 2 are scheduled and the goal is to calculate es_{10}. Since task 3 is the only unscheduled predecessor of task 10, es^2_{10} is not defined and es_{10} is calculated as follows:

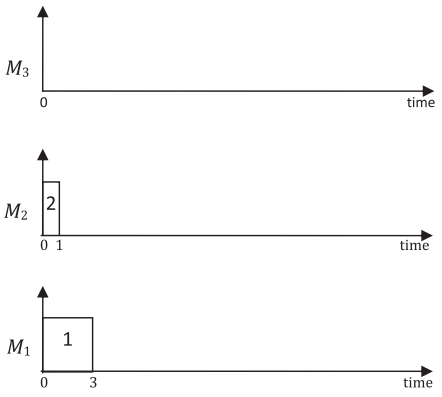

Fig. 4. Illustration of the lower bound in case (2).

$$es_{10}^1 = \min(\max(f_1, f_2 + \Delta_{2,10}), \max(f_2, f_1 + \Delta_{1,10})) = \min(\max(3, 1+1), \max(1, 3+5)) = 3,$$

$$es_{10}^3 = \min(f_1 + p_3, f_2 + p_3,\ \max(f_1, ef_3 + \Delta_{3,10}),\ \max(f_2, ef_3 + rDelta_{3,10}),\ \max(ef_3, f_1 + \Delta_{1,10}, f_2 + \Delta_{2,10}))$$
$$= \min(3+8, 1+8,\ \max(3, 8+3),\ \max(1, 8+3),\ \max(8, 3+5, 1+1)) = \min(11, 9, 11, 11, 8) = 8.$$

As a result, we have $es_{10} = \max(3, 8) = 8$ and consequently $ef_{10} = 17$

- *Case* 3 ($|pred^i_{v_u^l}| > m$): It is trivial that in this case, the predecessors of task i cannot all be processed in parallel and consequently some ones must precede some others. For each $t = 1, \ldots, T$, we represent by $PP^j_{v_u^l}(t)$ the profile of processor $M_j \in M$ corresponding to node v_u^l at time interval t ($[t-1, t)$). In case of having a task being processed on processor M_j at time interval t, we have $PP^j_{v_u^l}(t) = 1$, otherwise; $PP^j_{v_u^l}(t) = 0$. Moreover, for each node v_u^l, let α_j be the earliest possible start time of a task $k \in upred^i_{v_u^l}$ on processor M_j. Letting $p^{\min} = \min_{k \in upred^i_{v_u^l}} \{p_k\}$, α_j equals to the minimum time instant satisfying two conditions (21) and (22).

$$\alpha_j \geqslant \min_{k \in upred^i_{v_u^l}} \{es_k\}, \tag{21}$$

$$\sum_{t=\alpha_j}^{\alpha_j + p^{\min}} PP^j_{v_u^l}(t) = 0. \tag{22}$$

The first condition corresponds to the definition of the earliest start time while the second condition implies that processor M_j must be free consecutively, starting from time instant α_j for $p^{\min}$ time intervals. Assume β indicates the minimum completion time of all tasks of set $upred^i_{v_u^l}$. In order to calculate β, we use the idea developed for the lower bound of $Pm||C_{\max}$ (McNaughton [22]) in which it is assumed that the preemption is allowed and all tasks of set $upred^i_{v_u^l}$ are processed on those processors such that the minimum makespan is obtained for them. Thus, β equals to the maximum value satisfying (23).

$$\sum_{j=1}^{m} \sum_{t=\alpha_j}^{\beta} PP^j_{v_k^l}(t) \leqslant \sum_{k \in upred^i_{v_u^l}} p_k. \tag{23}$$

It should be noticed that β is identical for all processors and in the best case, all processors will be busy consequently until at least time instant β. Usually, if the communication delays are not much larger than the processing times, we can establish (24).

$$es_i^4 \geqslant \beta + \min_{k \in upred^i_{v_u^l}} \{\Delta_{ki}\}. \tag{24}$$

Also, if communication delays are much larger than the processing times, it may be better to schedule all tasks $k \in upred^i_{v_u^l}$ on the same processor, leading to (25).

$$es_i^4 = \min\left\{ \beta + \min_{k in upred^i_{v_u^l}} \{\Delta_{ki}\},\ \min_{j \in M}\{\alpha_j\} + \sum_{k \in upred^i_{v_u^l}} p_k \right\}. \tag{25}$$

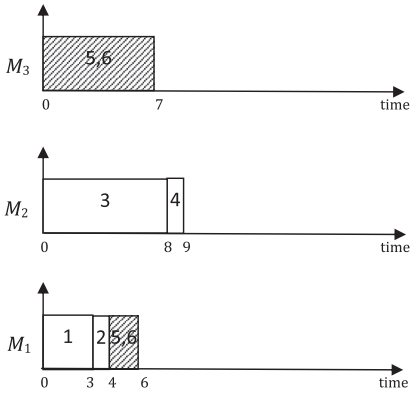

Fig. 5. Illustration of the lower bound in case (3).

On the other hand, since all arguments proposed in case (2) are applicable to this case, es_i could be developed as $es_i = \max\{es_i^1, es_i^2, es_i^3, es_i^4\}$.

As an example, consider the partial schedule depicted in Fig. 5 in which tasks 1–4 are scheduled while we desire to calculate es_9. Here, we have $\alpha_1 = 5,\ \alpha_2 = 10,\ \alpha_3 = 0$ and $\beta = 7$ where $\beta = 7$ has been shown in Fig. 5 by dashed boxes. Upon calculating $es_9^1 = 4,\ es_9^2 = 6,\ es_9^3 = 6$ and $es_9^4 = 8$, we get $es_9 = \max\{es_9^1,\ es_9^2,\ es_9^3,\ es_9^4\} = 8$.

3.2.3. Dominance rules

In this section, three dominance rules are developed. The first two rules are proposed to prevent the creation of repetitive schedules, generated due to the branching scheme. Demeulemeester and Herroelen [23] and Demeulemeester et al. [24] have developed such rules, referred to as cut set rules, in their developed B&B algorithms. Also, the third rule is constructed to avoid low quality schedules. In each node of the B&B tree, these dominance rules are applied consecutively. In other words, if a node is not fathomed by the dominance rule 1, the second dominance rule is applied and so forth.

Definition. Two tasks i and k are called independent, if there is no path between them in directed graph $G(N,A)$.

Dominance rule 1. Consider two nodes v_u^l and $v_{u'}^{l'}$ corresponding to assignments $i-j$ and $i'-j'$, respectively. Also, assume $l' = l+1$ and there is a path between nodes v_u^l and $v_{u'}^{l'}$ in the B&B tree. Node $v_{u'}^{l'}$ is fathomed, if the following conditions are satisfied:

(I) Tasks i and i' are independent,
(II) $i > i'$.

Proof. Assume node $v_{u'}^{l'}$ corresponds to a partial schedule $\boldsymbol{s}$. The first condition implies that if two nodes v_u^l and $v_{u'}^{l'}$ are exchanged in the B&B tree, the partial schedule $\boldsymbol{s}$ is obtained again. Consequently, in such condition, we must consider either assignment $i-j$ before assignment $i'-j'$ or vice versa. Since, $i > i'$, we prefer to investigate the former case. □

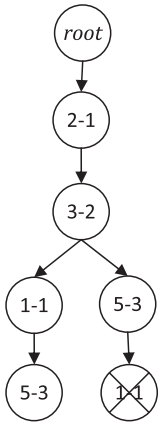

Fig. 6. Illustration example for *dominance rule* 1.

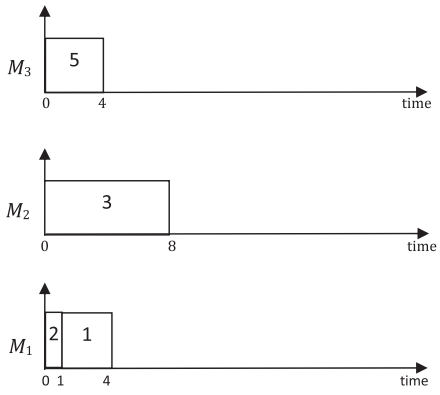

Fig. 7. Partial schedule corresponding to Fig. 6.

For example, consider the B&B tree illustrated in Fig. 6, in which two different paths are displayed where both correspond to the partial schedule depicted in Fig. 7.

Based on *dominance rule* 1, since tasks 1 and 5 are independent and they are assigned to two differ processors M_1 and M_3, respectively, the right path in which assignment 1–1 is investigated after assignment 5–3 is fathomed.

Dominance rule 2. Consider two nodes v_u^l and $v_{u'}^l$ corresponding to assignments $i-j$ and $i-j'(j'<j)$, respectively. If v_u^l and $v_{u'}^l$ have the same parent at level $l-1$ and $s_i = s'_{i'}$, fathom node $v_{u'}^l$ (s_i and s'_i correspond to assignments $i-j$ and $i-j'$, respectively).

Proof. Since nodes v_u^l and $v_{u'}^l$ have the same parent at level $l-1$, for each $t=1,\ldots,T$, before doing assignments $i-j$ and $i-j'$, $PP^j_{v_u^l}(t)$ and $PP^{j'}_{v_{u'}^l}(t)$ are identical. Thus, for each offspring of node $v_{u'}^l$ such as assignment $k-j''$ and its corresponding partial schedule $\boldsymbol{s}$, there is an identical assignment among offspring of node v_u^l which corresponds exactly to the partial schedule $\boldsymbol{s}$. □

As an example, consider the B&B tree illustrated in Fig. 8 and its corresponding partial schedule, depicted in Fig. 9. It is obvious that scheduling of task 3 on processor M_2 or M_3 is identical because these two processors are identical.

Dominance rule 3. In each node of the B&B tree, if $UB \leqslant LB$, fathom the node.

Proof. Straightforward. □

It should be mentioned that when using *dominance rule* 3, we first check the condition $UB \leqslant LB$[1] and in case of violating, we then check whether $UB \leqslant LB$ or not.

4. Results and discussion

The B&B algorithm is implemented in Visual C++ 2010 and all experiments were run on a Core i5 2.4 GHz Pentium IV laptop computer with 4 GB of RAM, equipped with Windows 7 Ultimate. The maximum CPU usage for running the program is restricted to 25% (1 Core) and the maximum memory size for storing the tree structure is restricted to 1 MB.

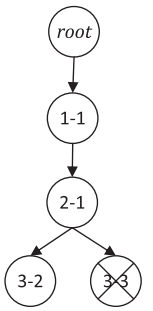

Fig. 8. Illustration example for *dominance rule* 2.

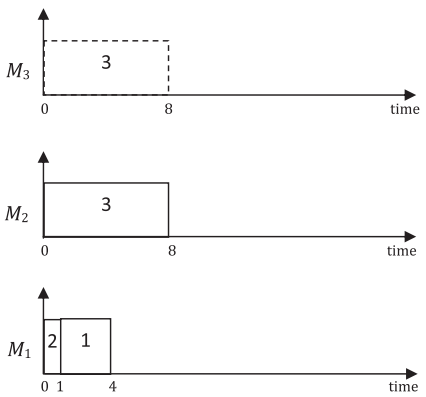

Fig. 9. Partial schedule corresponding to Fig. 8.

4.1. Experimental setup

Since there is no benchmark instance, available in the literature for this problem, we generated a set of random test problems using the random network generator RanGen (Demeulemeester et al. [25]). For this purpose, a full factorial design based on various values of the three following parameters is developed: number of tasks ($n = 12, 14, 16$), number of processors ($m = 2, 3, 4$) and order strength ($OS = 0.4, 0.6, 0.8$). For each combination of n, m and OS, five random test problems are generated, resulting in 135 test problems. Moreover, processing times, communication delays and weights of the tasks are realizations of independent discrete uniform random variables on the interval [1,10]. The due date d_i of each task $i \in N$ is a realization of normal random variables $N[\mu_i,\ \sigma_i^2]$ where μ_i and σ_i are determined as follows. If $|Pred_i| = 0$, then $\mu_i = ef_i$; else if $|Pred_i| = 1$, then $\mu_i = \mu_{pred(i)} + p_i$; otherwise, $\mu_i = ef_i + Avg_{comu}$ in which Avg_{comu} gives the average values of communication delays which is set to 5 for the generated test instances. The parameter σ_i is determined such that f_i is less than ef_i with probability of approximately zero.

For this purpose, we consider $\sigma_i = \frac{\mu_i - ef_i}{\sqrt{10}}$. Also, it should be noticed that, since generated due dates may not be integer, their rounded values are considered as due dates.

In order to evaluate the impact of processing times and communication delays on the quality of the obtained solutions, we consider uniform random variables on the interval [1,10] and [11,20] as small and large values, respectively. Thus, we refer to the abovementioned test set as TS^{SS} in which both processing times and communication delays are small. We generated there other test sets, referred to as TS^{SL}, TS^{LS} and TS^{LL} where TS^{SL} indicates the test set in which processing times are small and communication delays are large.

4.2. Summary results

In this section, the total CPU run time of the B&B algorithm is referred to as T_{Total} and is expressed in seconds. Table 3 represents T_{Total} for the full factorial experiments, constructed based on parameters n, m and OS. Each cell of this table represents the average T_{Total} for the corresponding five test problems of TS^{LL} while in the last row and column, the average T_{Total} for different values of n and OS are reported, respectively. Moreover, based on the information reported in this table, the average T_{Total} for $m = 2, 3$ and 4 is 87.09, 318.09 and 372.59 s, respectively. Also, the total average of all experiments is 259.26 s. As it was expected, the CPU run time is increased by increasing the values of n, m and decreasing the value of OS.

It should be noticed that both processing times and communication delays affect the performance of the B&B algorithm such that the average T_{Total} is increased for the three other test sets. The results reported in Table 4 indicate that communication delays have larger impact on the average T_{Total} rather than processing times.

4.3. Comparative results with time limits

In order to evaluate the performance of the developed B&B algorithm, we compare it with ILOG CPLEX solver 12.3. Based upon the formulation developed in Section 2.3, we compare the performance of the B&B algorithm and CPLEX solver based upon four test sets TS^{SS}, TS^{SL}, TS^{LS} and TS^{LL} and for the time limits $TL = 1, 10, 30, 60$ and 100 s. Each triple reported in Table 5 gives, respectively, the average percentage of deviation (*APD*) of the best found solutions from the optimal ones, the number of optimal solutions found (*#opt*) and total number of test instances for which no feasible solution exist, within the corresponding time limits. Since the CPLEX could not find any feasible solution for some test problems within some given time limits, *APD* and *#opt* are reported based on those test instances for which at least a feasible solution has been found. It should be noticed that the B&B algorithm has been able to find feasible solution for all test instances even for $TL = 1$ s.

The results of Table 5 indicate that our developed B&B algorithm outperforms CPLEX especially for smaller time limits. Our developed B&B algorithm is efficient such that it is able to find the optimal solutions of around (in average) 79% of

Table 3
The average T_{Total} for different values of n, m and OS.

n	OS									
	0.4			0.6			0.8			
	m			m			m			Avg.
	2	3	4	2	3	4	2	3	4	
12	0.90	1.83	1.75	0.10	0.34	0.74	0.02	0.01	0.02	0.55
14	14.57	21.31	88.92	1.28	6.34	22.76	0.00	0.02	0.03	17.25
16	739.96	2154.53	2364.40	26.95	678.93	873.62	0.01	0.27	1.07	759.97
Avg.	598.60			198.54			0.16			259.26

Table 4
The average T_{Total} for test sets TS^{SS}, TS^{SL}, TS^{LS} and TS^{LL}.

TS^{SS}	TS^{SL}	TS^{LS}	TS^{LL}
54.60	236.75	79.07	259.26

Table 5
Comparing the performance of the B&B algorithm and CPLEX in limited times.

Test set	*TL*=	1	10	30	60	100
TS^{SS}	B&B	3.26, 111, 0	1.66, 124, 0	1.09, 128, 0	1.04, 129, 0	0.73, 130, 0
	CPLEX	–, 0, 135	76.15, 9, 93	58.73, 30, 62	45.31, 57, 36	32.78, 77, 19
TS^{SL}	B&B	5.70, 103, 0	1.48, 117, 0	1.13, 122, 0	0.91, 125, 0	0.60, 128, 0
	CPLEX	–, 0, 135	77.85, 4, 108	45.48, 38, 80	30.04, 40, 53	25.31, 53, 42
TS^{LS}	B&B	5.53, 113, 0	1.31, 125, 0	0.86, 127, 0	0.71, 127, 0	0.53, 128, 0
	CPLEX	–, 0, 135	83.25, 0, 134	81.31, 0, 117	89.28, 1, 109	75.35, 4, 92
TS^{LL}	B&B	2.73, 102, 0	1.32, 119, 0	0.86, 124, 0	0.42, 127, 0	0.41, 128, 0
	CPLEX	–, 0, 135	–, 0, 135	47.26, 0, 127	60.61, 0, 116	49.82, 1, 107

Table 6
Impact of the communication delays on the performance of the initial solution.

TS^{SS}	TS^{SL}	TS^{LS}	TS^{LL}
19.43	26.14	13.35	7.41

the test instances in at most one second while CPLEX has not been able to find any optimal solutions in this time limit. This percentage is increased to around 95% and 25% for B&B and CPLEX, respectively, when *TL* = 100 s.

4.4. Impact of the processing times and communication delays on the performance of the initial solution

In order to evaluate the impact of the processing times and communication delays on the performance of the initial solution, we consider the average percent deviation of the solutions obtained by the initialization procedure, from the optimal solutions for four test sets TS^{SS}, TS^{SL}, TS^{LS} and TS^{LL}. The results represented in Table 6 indicate that the performance of the initial solution will be increased when processing times are increased. As it was expected, increasing the communication delays has a negative impact on the efficiency of the initial solutions, since the communication delays are not considered in calculation of θ_i values.

5. Conclusions

This article has studied a model for scheduling a set of precedence-dependent tasks on a set of identical processors while communication delays are imposed between tasks due to the data transferring by processing two directly dependent tasks on different processors. An integer linear model and an efficient B&B algorithm were developed for $P_m|prec, comu|Y_w$. Computational performance of the developed B&B algorithm has been examined. Also, the developed integer linear model has been solved by ILOG CPLEX 12.3 and computational results indicate the superiority of our developed B&B algorithm, especially for small time limits.

Developing more sophisticated exact or (meta) heuristic solution techniques could be interesting research topics. Furthermore, developing smaller valid upper bound T for the makespan of the optimal solution of $P_m|prec,\ comu|Y_w$, can increase noticeably the efficiency of the developed model. Finally, as a more fundamental extension, it could be assumed that the communication delays are dependent to the location of processors.

Appendix A

Remark. There is an optimal solution for $P_m|prec,\ comu|Y_w$ with the makespan less than or equal to $T = \sum_{i=1}^{n} p_i$.

Proof. Consider the schedule ***s*** in which all tasks are sequentially processed with respect to precedence relations on the same processor with the makespan $\sum_{i=1}^{n} p_i$. Also, assume this makespan is not a valid upper bound for the makespan of the optimal solution of $P_m|prec,\ comu|Y_w$. In order to construct a valid upper bound for the makespan of the optimal solution, we need to construct a schedule with the makespan larger than $\sum_{i=1}^{n} p_i$. For this purpose, we need to postpone at least one task in schedule **s** but this makes the late work of this job worse and consequently the obtained schedule cannot be optimal. Thus, $T = \sum_{i=1}^{n} p_i$ is a valid upper bound for the makespan of the optimal solution of $P_m|prec,\ comu|Y_w$. □

References

[1] M.O. Tokhi, M.A. Hossain, Homogeneous and heterogeneous parallel architectures in real-time signal processing and control, Control Eng. Pract. 3 (12) (1995) 1675–1686.

[2] H.B. Prajapati, S.K. Vij, Analytical study of parallel and distributed image processing, in: Paper Presented at the in Proceeding of the 2011 International Conference on Image Information Processing (ICIIP), 2011.
[3] M.C. Jadud, C.L. Jacobsen, C.G. Ritson, J. Simpson, Safe parallelism for robotic control, in: IEEE International Conference on Technologies for Practical Robot Applications, 2008, pp. 137–142.
[4] A. Arisoy, M.K. Bayrakceken, S.G.M. Basturk, O.S. Bogosyan, High order sliding mode control of a space robot manipulator, in: 5th International Conference on Recent Advances in Space Technologies, 2011, pp. 833–838.
[5] O. Sinnen, Task Scheduling for Parallel Systems, John Wiley & Sons Inc, Hoboken, New Jersey, 2007.
[6] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling theory: a survey, Ann. Discrete Math. 5 (1979) 287–326.
[7] B. Veltman, B.J. Lageweg, J.K. Lenstra, Multiprocessor scheduling with communication delays, Parallel Comput. 16 (2–3) (1990) 173–182.
[8] M. Gendreau, G. Laporte, E.M. Guimarães, A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times, Eur. J. Oper. Res. 133 (1) (2001) 183–189.
[9] J. Blazewicz, P. Dell'Olmo, M. Drozdowski, P. Mączka, Scheduling multiprocessor tasks on parallel processors with limited availability, Eur. J. Oper. Res. 149 (2) (2003) 377–389.
[10] J. Verrite, Scheduling tree like task systems with uniform deadlines subject to unit-length communication delays, Discrete Appl. Math. 101 (2000) 269–289.
[11] M. Drozdowski, Scheduling for Parallel Processing, Springer, London, 2009.
[12] C.N. Potts, L.N. Van Wassenhove, Approximation algorithms for scheduling a single machine to minimize total late work, Oper. Res. Lett. 11 (5) (1991) 261–266.
[13] M.Y. Kovalyov, C.N. Potts, L.N. Van Wassenhove, A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work, Math. Oper. Res. 19 (1) (1994) 86–93.
[14] J. Blazewicz, Scheduling preemptible tasks on parallel processors with information loss, Tech. Sci. Inf. 3 (6) (1984) 415–420.
[15] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, Flow shop scheduling with late work criterion choosing the best solution strategy, Lect. Notes Comput. Sci. 3285 (2004) 68–75.
[16] E. Pesch, M. Sterna, Late work minimization in flow shop by a genetic algorithm, Comput. Ind. Eng. 57 (4) (2009) 1202–1209.
[17] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, A note on two-machine job shop with weighted latework criterion, J. Sched. 10 (2) (2007) 87–95.
[18] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, Open shop scheduling problems with late work criteria, Discrete Appl. Math. 134 (1) (2004) 1–24.
[19] M. Sterna, A survey of scheduling problems with late work criteria, Omega 39 (2011) 120–129.
[20] M. Sterna, Problems and Algorithms in Non-Classical Shop Scheduling, Scientific Publishers of the Polish Academy of Sciences, Poznan, 2000.
[21] J.E.J. Kelley, M.R. Walker, Critical path planning and scheduling, in: Eastern Joint IRE-AIEE-ACM Computer Conference, vol. 16, 1959, pp. 160–172.
[22] R. McNaughton, Scheduling with deadline and loss function, Manage. Sci. 6 (1959) 1–12.
[23] E.L. Demeulemeester, W.S. Herroelen, A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, Manage. Sci. 38 (1992) 1803–1818.
[24] E. Demeulemeester, B. De Reyck, W. Herroelen, The discrete time/resource trade-off problem in project networks: a branch-and-bound approach, IIE Trans. 32 (2000) 1059–1069.
[25] E. Demeulemeester, M. Vanhoucke, W. Herroelen, A random generator for activity-on-the-node networks, J. Sched. 6 (2003) 13–34.